

The Bug Shop

Ben Nagy & The Grugq



Overall Idea

1. Find Bugs
2. Exploit Bugs
3. Sell Bugs
4. Profit!



LESSONS LEARNED THE HARD WAY

“FUZZING” IS A VAGUE TERM

Target Selection



Fetching



Tracing

Template Selection



Case Generation



Delivery

Instrumentation



Triage



Repro

Distill



Root Cause

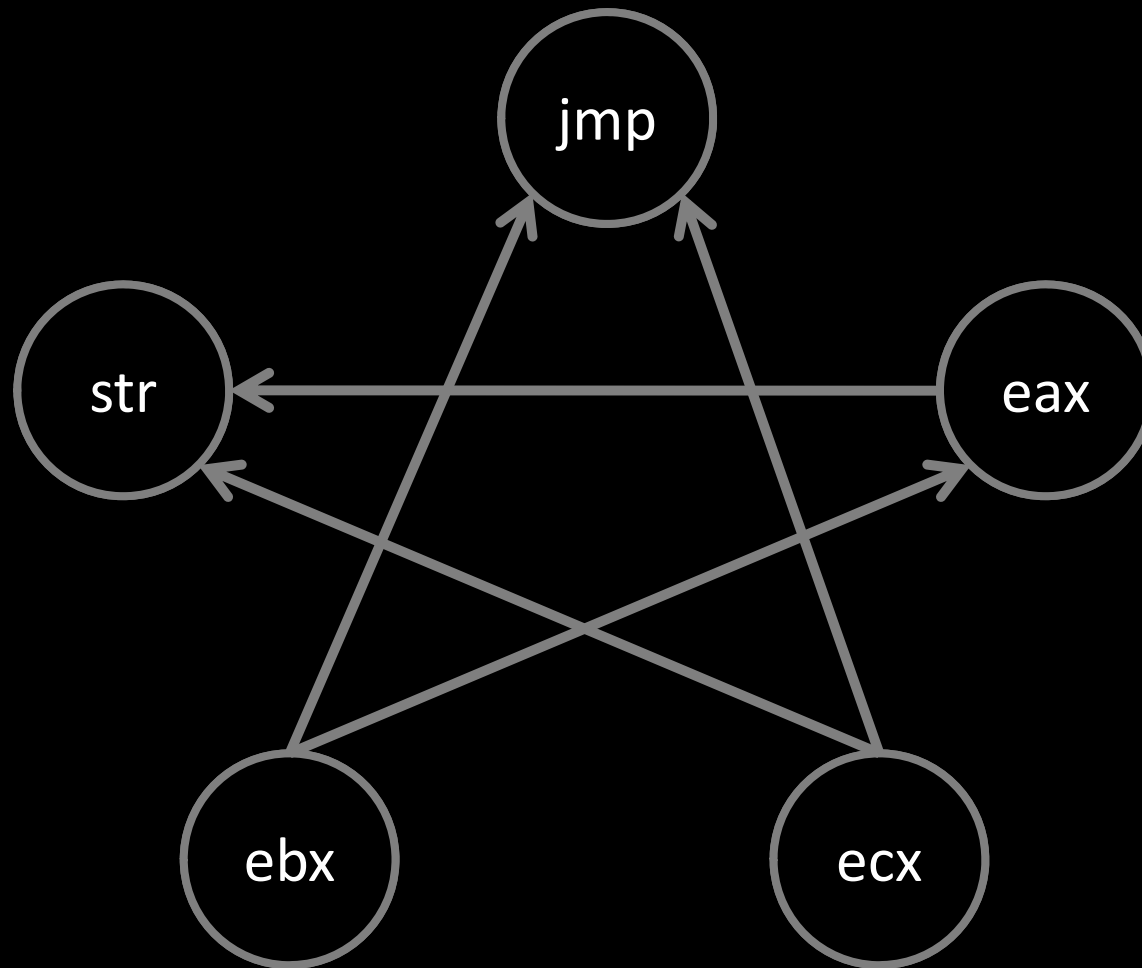


Exploit



Sell!

Hardw4reZ



Hardw4reZ

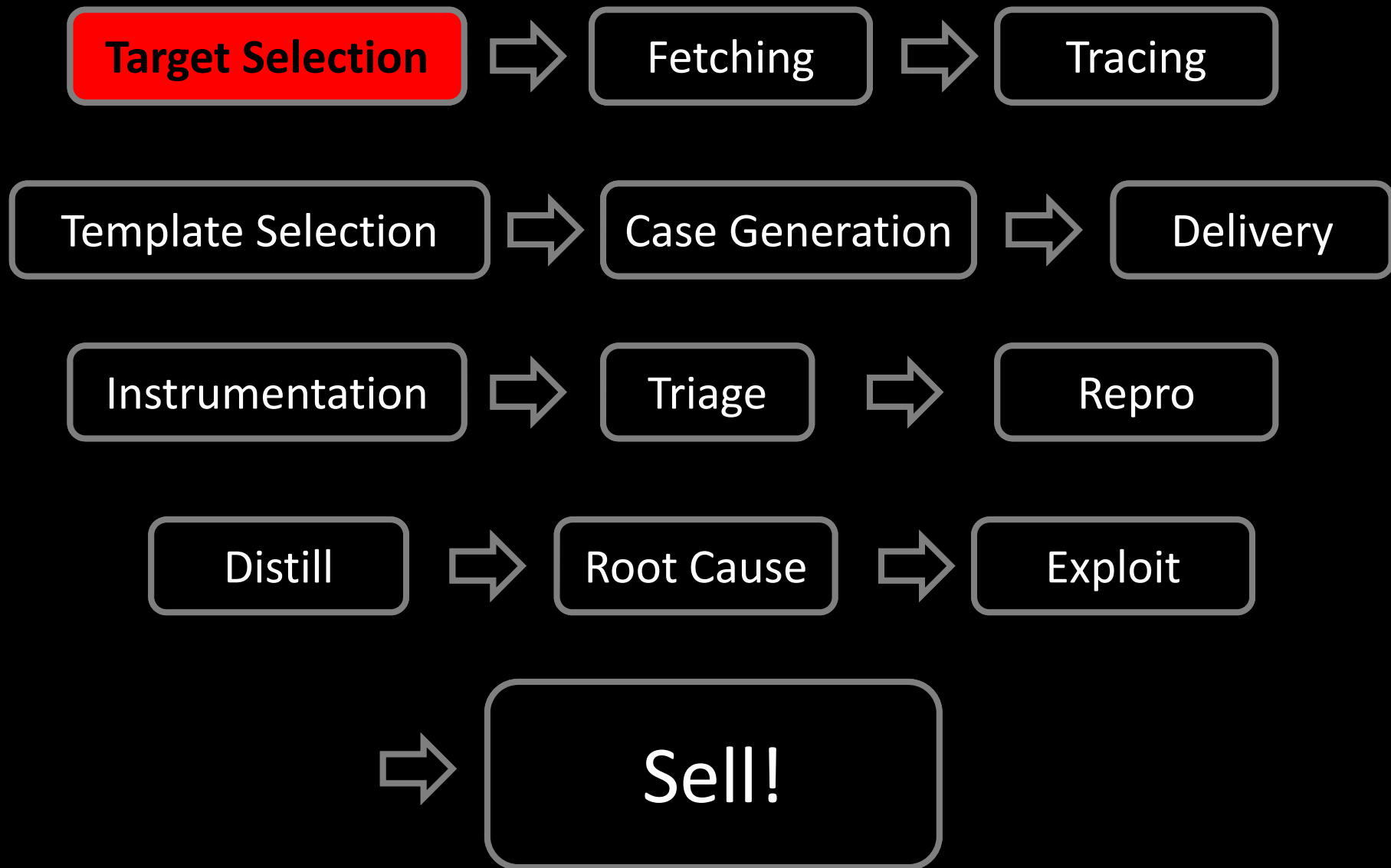


Hardw4reZ

- Virtualise with KVM
- 48 core VM hosts (4 x 12 core AMD Opteron)
- 20Gb Infiniband fabric
- Shared storage via... NFSv4! It's back!
- Scripty awesomeness for management
- \$280 USD per core / \$140 per VM
- Pulls ~10kW

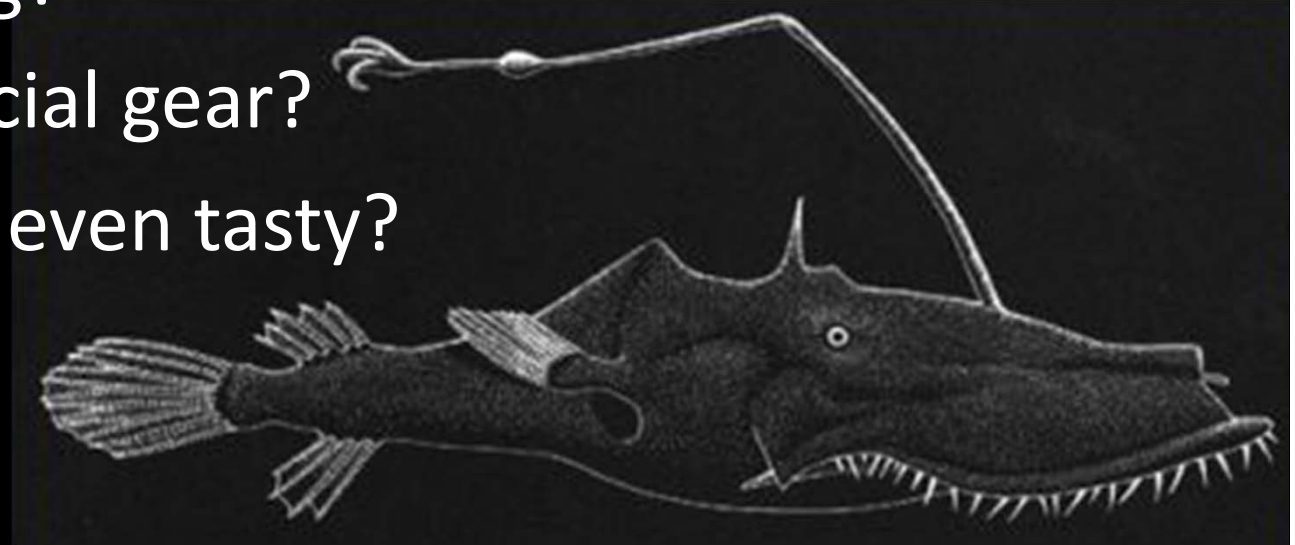
LESSONS LEARNED THE HARD WAY

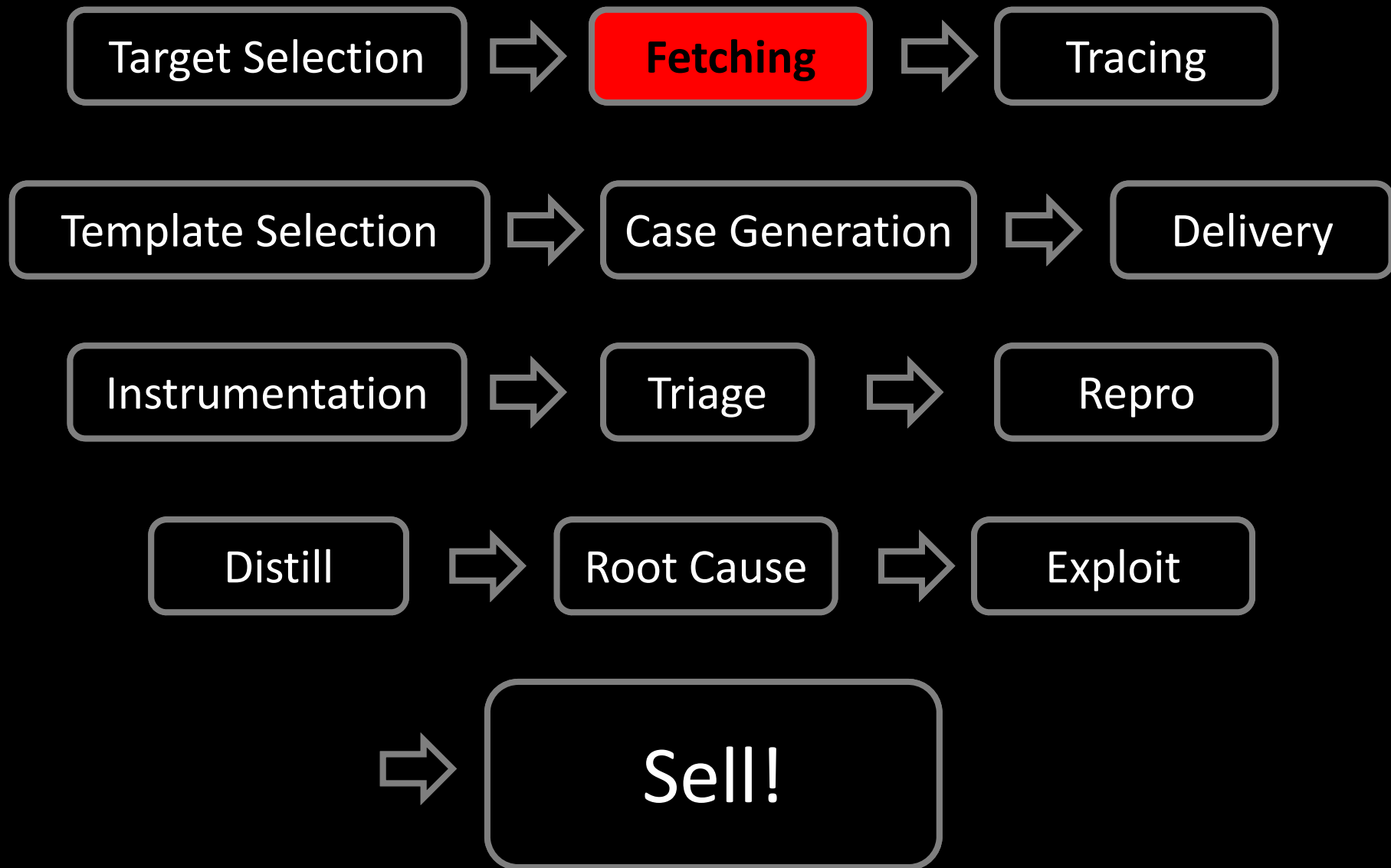
LINUX KERNEL SUCKS



Target Selection

- Are there fish?
- Who else is fishing here?
- Are they biting?
- Do I need special gear?
- Are these fish even tasty?



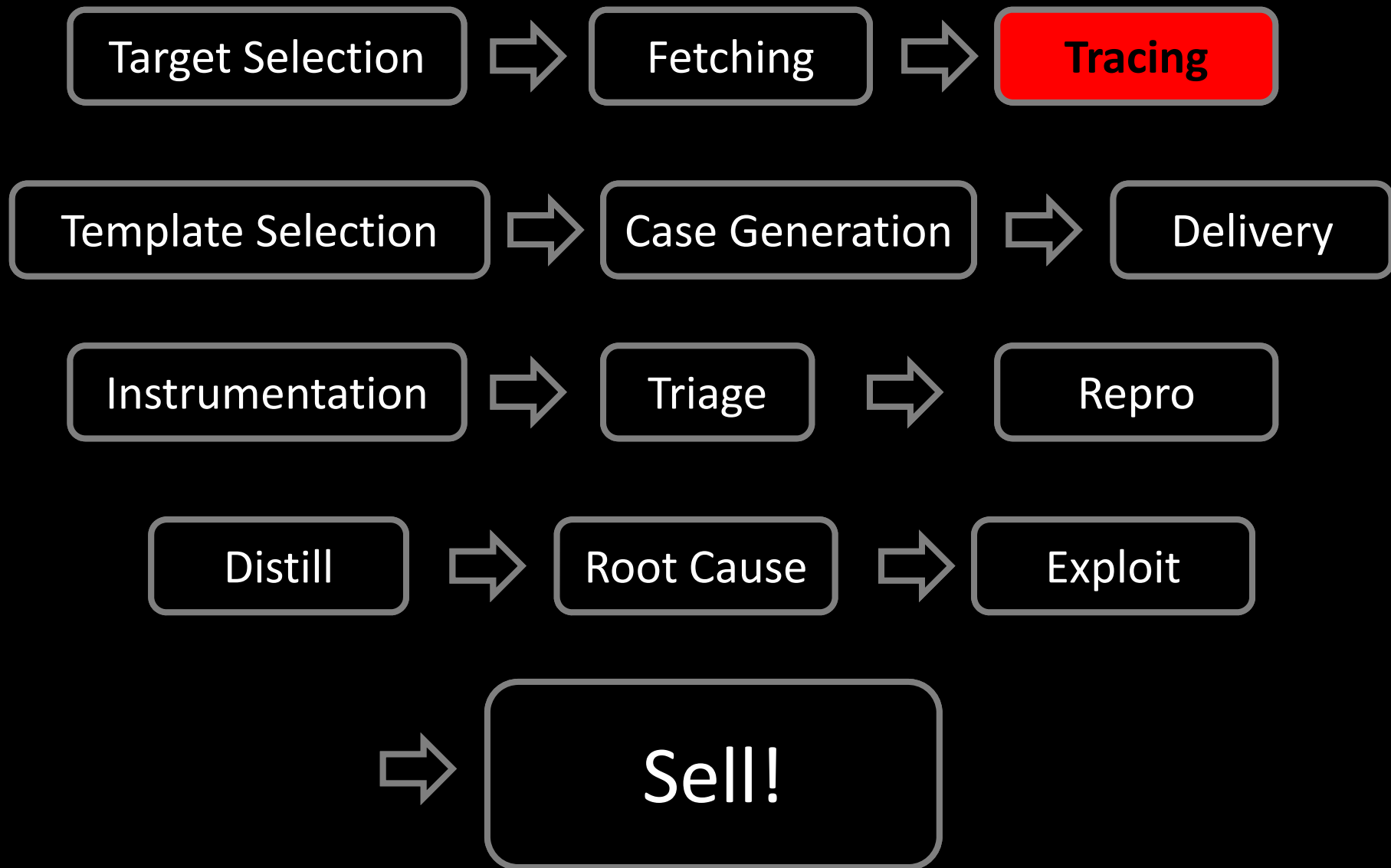


fetching.py

- Asynchronous, massively parallel downloader
- Many features, very versatile
- Seriously scaleable
- Based on Python gevent
 - Wrapper for greenlets / libevent
 - Transparently asynchronous

- Download here:

<https://github.com/grugq/prospector>



LESSONS LEARNED THE HARD WAY

STOP SCREWING AROUND. USE PIN.

ccovtrace.dll

- Dynamic instrumentation – gets ALL blocks
- We trace ALL basic blocks and then whitelist
 - We're tracking edges <addr> → <addr> not nodes
 - Can't get 'half' edges without tracing everything
- Records edge 'weight' (times travelled)
 - which we don't use yet

ccovtrace.dll

- Anything we don't care about is a raw address
 - converted to an OUTSIDE index in postprocessing
- Anything we do care about is mod+0xffset
- Simple textfile output

ccovtrace.dll

- Very much not rocket science
- “Fast Enough” – 2-5+ minutes to trace Word
- “Stable Enough” – no faults in 100k+ traces
- Download here:

<https://github.com/grugq/RunTracer>

Target Selection



Fetching



Tracing

Template Selection



Case Generation



Delivery

Instrumentation



Triage



Repro

Distill



Root Cause



Exploit



Sell!

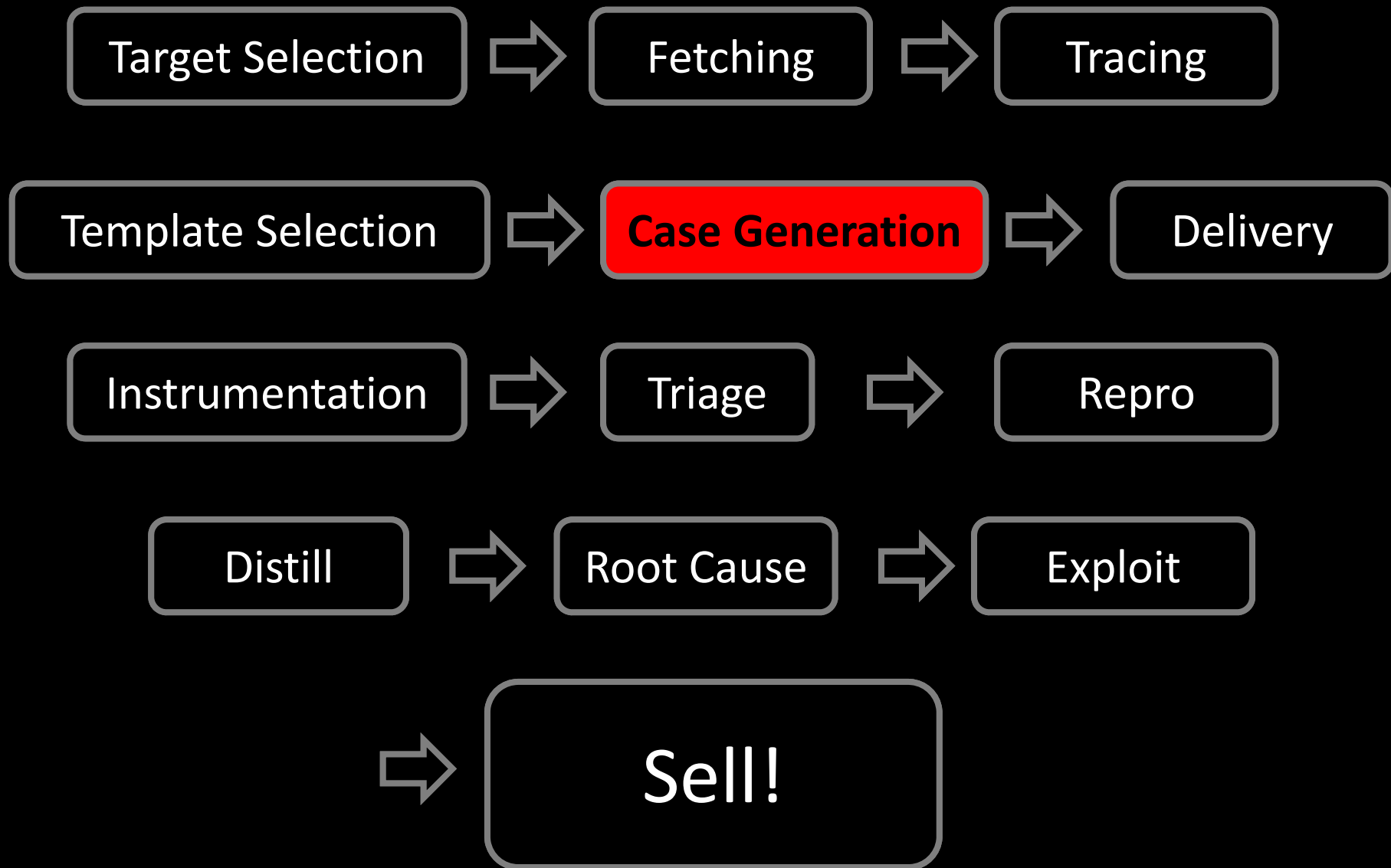


Compsci is Hard! ... Let's go Shopping!

Template Selection

- Apply Set Cover algorithm(s) to traces
- Minimize files, maximize coverage
- For more detail:

<http://www.ruxcon.org.au/assets/Presentations/ben-nagy.prospecting-for-rootite.2010.pdf>



LESSONS LEARNED THE HARD WAY

YOU HAVE A FANCY CASE GENERATOR?!

LESSONS LEARNED THE HARD WAY

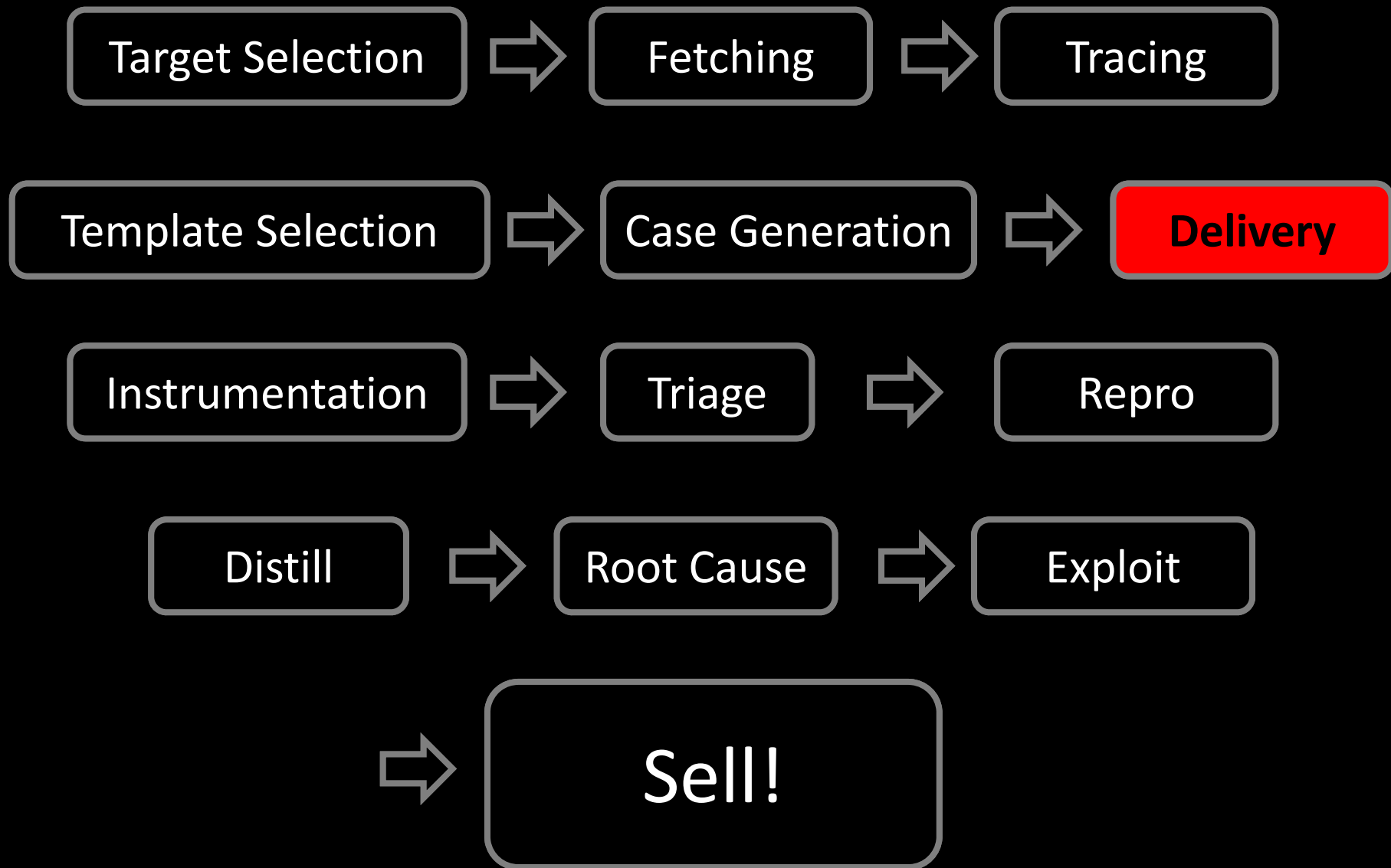
**YOU HAVE A FANCY CASE GENERATOR?!
NOBODY CARES.**

LESSONS LEARNED THE HARD WAY

YOU HAVE A FANCY CASE GENERATOR?!

NOBODY CARES.

(UNLESS IT'S SAGE)



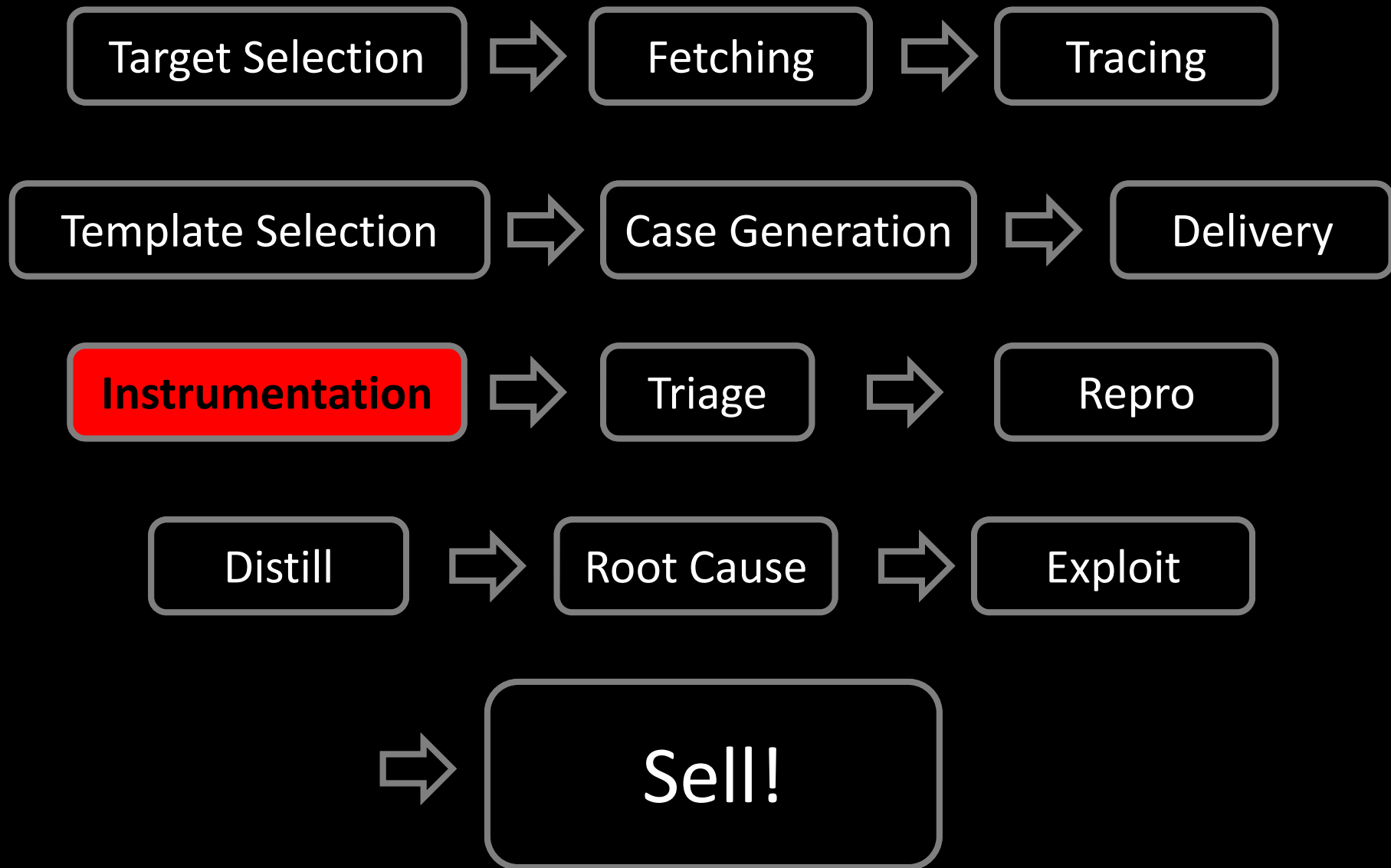
LESSONS LEARNED THE HARD WAY

FLEXIBILITY, RELIABILITY, SCALABILITY

(DO NOT HAPPEN BY ACCIDENT)

tag files

```
producer_filename:/mnt/nfs/raid/crashes-jan2011/4E4A2C8D-23CB-  
11E0-E392-559F3CE4B21D.doc  
producer_crc32:121ac32f  
producer_timestamp:2011-04-13 15:08:10 +0800  
producer_iteration:21911  
producer_id:9DEEE08E-5422-4674-BEF0-280576714834  
fuzzbot_delivery_options:{}  
fuzzbot_extension:doc  
fuzzbot_command:C:\Program Files\Microsoft  
Office\Office11\WINWORD.EXE /Q  
fuzzbot_timestamp:2011-04-13 15:07:57 -0700  
fuzzbot_delivery_time:1.15625  
fuzzbot_crash_md5:895a3fcb76691b5d1ade97bf5f49b775  
fuzzbot_exception_info_md5:18a1126766a0c8aef134d0e41d9bd759  
fuzzbot_crash_crc32:121ac32f  
fuzzbot_crash_uuid:35727CC4-661A-11E0-1AAB-B6E168FFE63B
```



(r)Buggery

- Wrapper(s) for dbgeng.dll (aka WinDbg)
- One in Python (replaces pydbgeng)
- One in Ruby (first one, afaik)
- Script anything you can do with WinDbg
 - Full access to all extensions (!exploitable, !avrf, !heap)
 - Crazy-ass awesome WinDbg conditional breakpoints
 - Execute commands (“u @eip”) or use the native API

rBuggery

- Why did nobody do this before? COM is weird.
 - No ComTypes for Ruby.
- OutputCallback, EventCallbacks
- As usual I cheated.
- Build the COM objects raw in memory...

```
@AddRef = Win32::API::Callback.new('P','L') {|p| 1 }  
# ... then  
# pack the callback addresses as an array of uint32 (pointers)  
# and then use the 'P' pack directive to get a pointer to  
# that string.  
@iface_ptr=[@vtable.map {|cb| cb.address }.pack('L*')].pack('P')
```

rBuggery (demo)

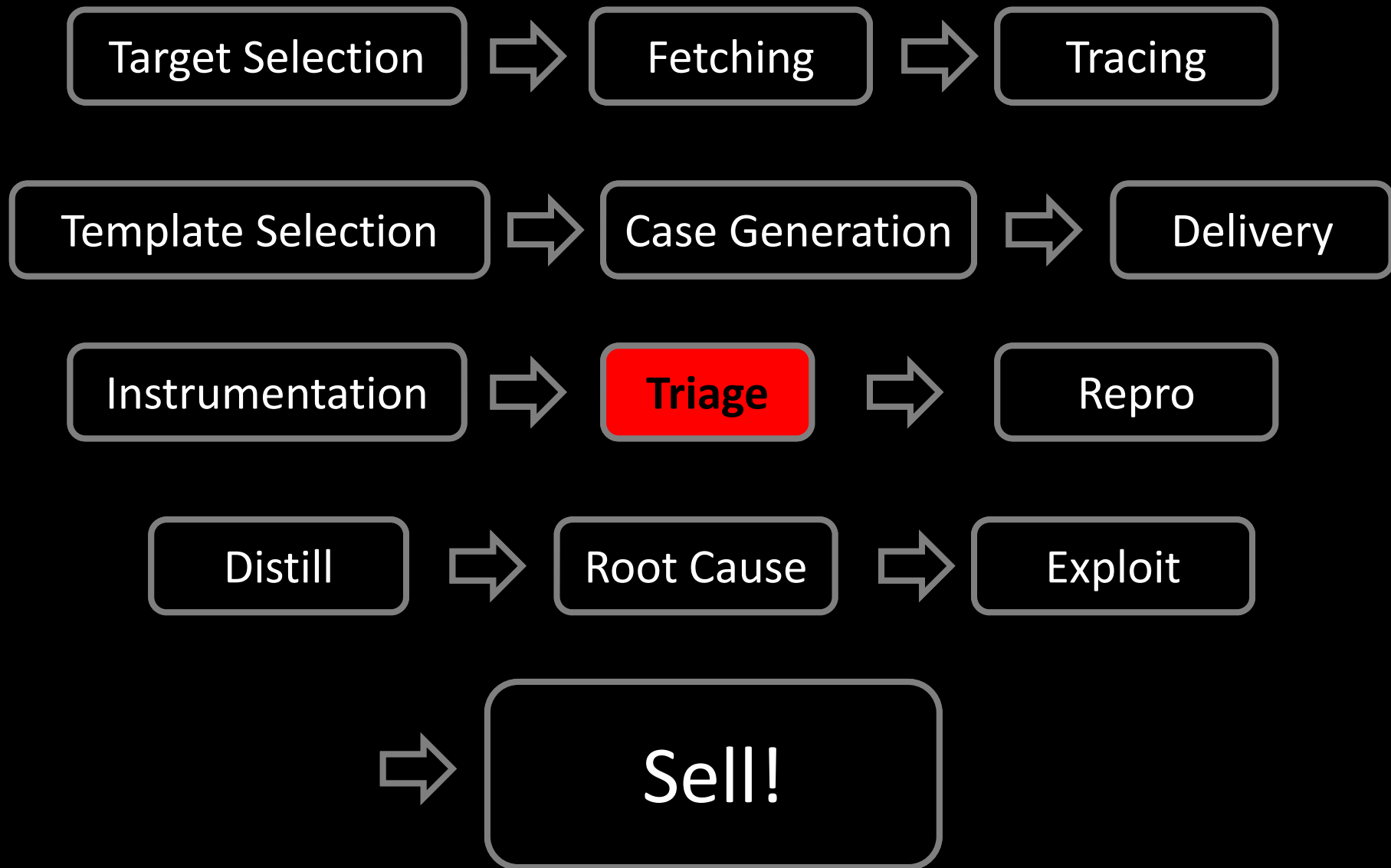
```
debug_client.create_process("notepad.exe")
debug_client.execute ".symopt+0x100" # NO_UNQUALIFIED_LOADS
debug_client.execute ".sympath C:\\\\localsymbols"
debug_client.break
debug_client.wait_for_event( -1 )
type, desc, extra=debug_client.get_last_event_information
puts debug_client.exception_record
puts debug_client.execute 'r'
puts debug_client.disassemble(
    debug_client.registers['eip'], 10
).map {|a| a.join(' ')}
debug_client.terminate_process
```

(r)Buggery

- Download here:

<https://github.com/grugq/Buggery>

<https://github.com/bnagy/rBuggery>



LESSONS LEARNED THE HARD WAY

**ANNOYING, HARD TO READ,
GREEN SCREEN DEMOS NEVER
GET OLD!**

Target Selection



Fetching



Tracing

Template Selection



Case Generation



Delivery

Instrumentation



Triage



Repro

Distill



Root Cause

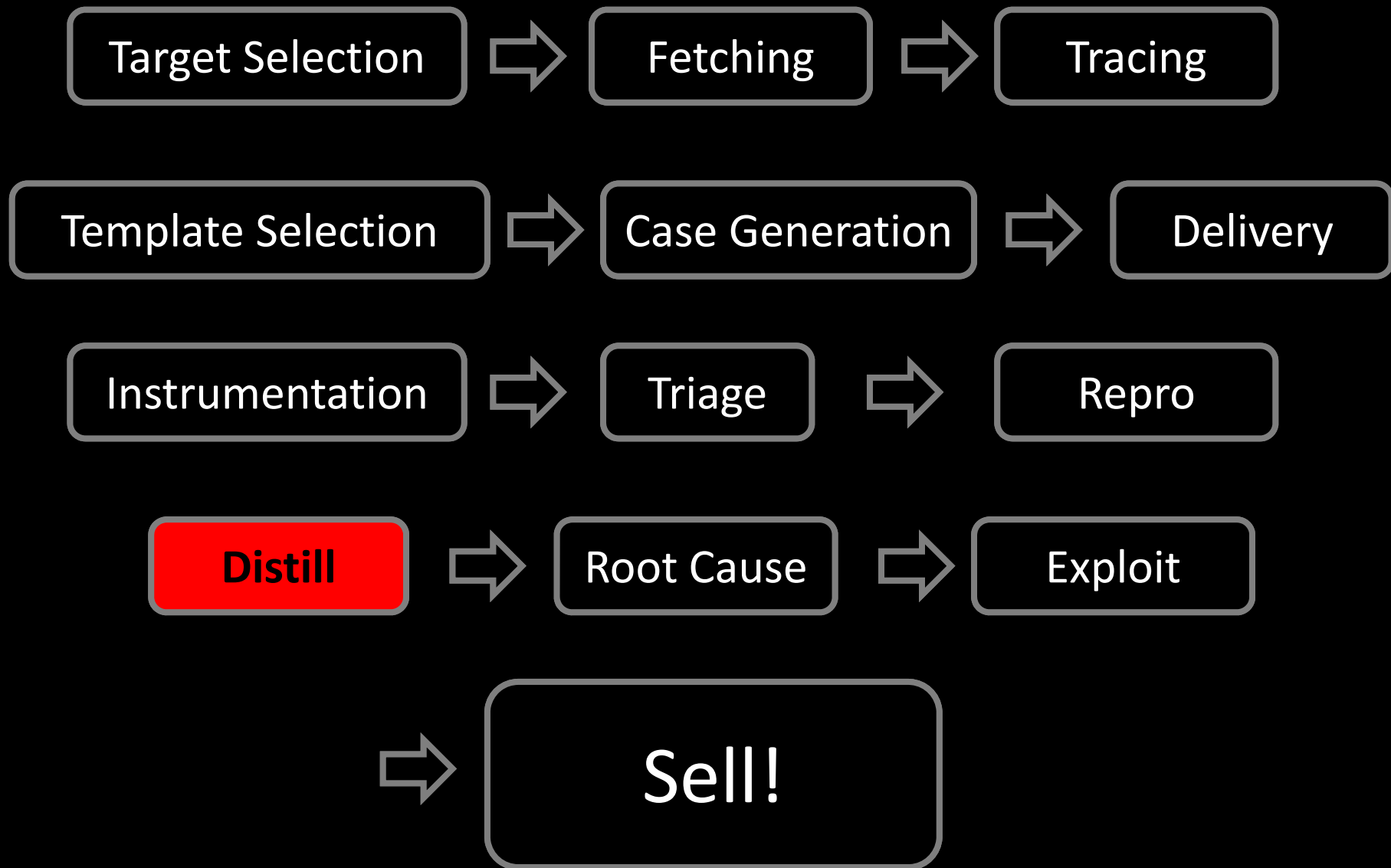


Exploit



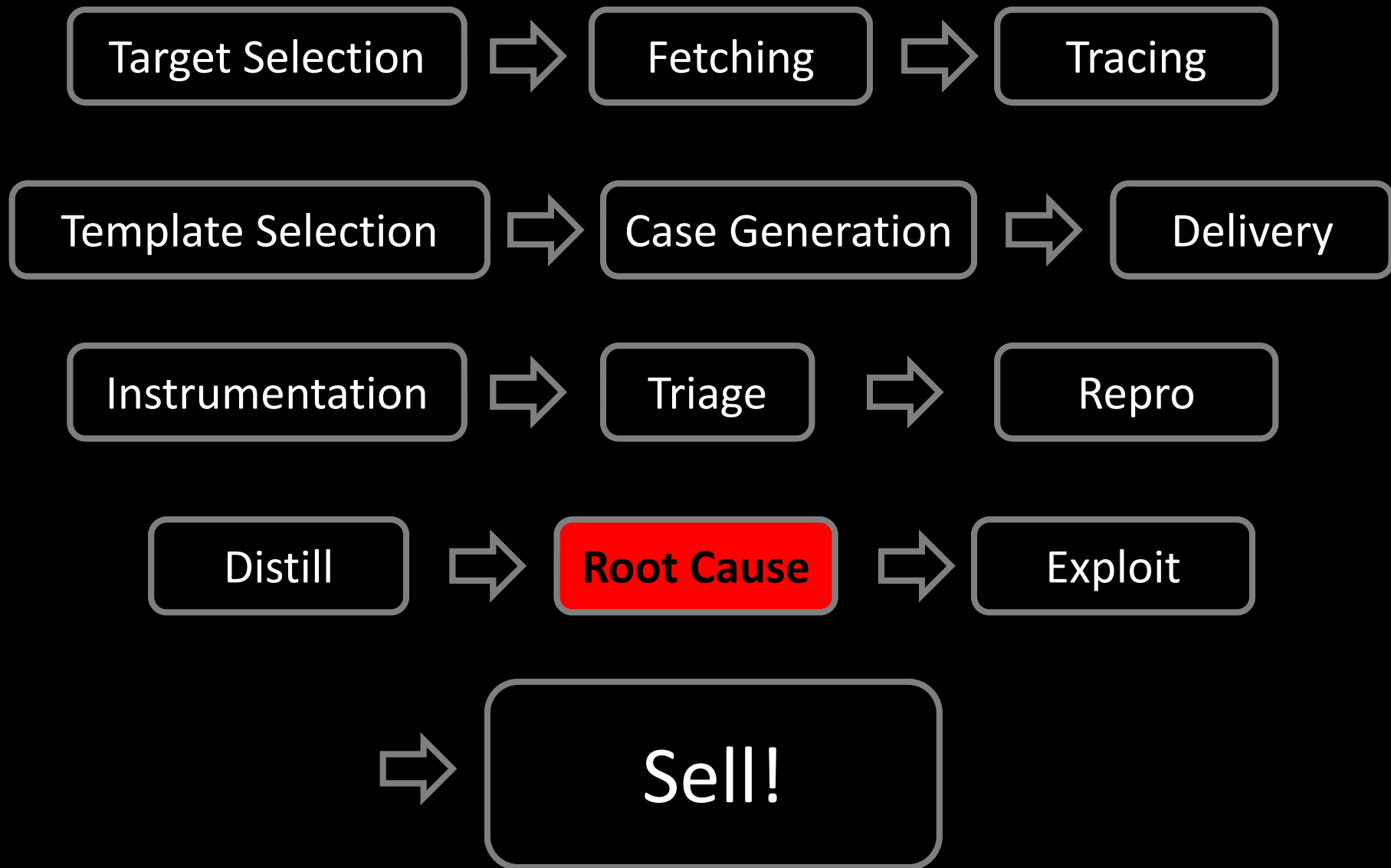
Sell!





Distill

- “Shears” is a better name, but it’s taken
- Revert a mutated file, byte by byte, test for crash
- Sounds simple?
 - Different crashes
 - Hang vs crash?
- Currently crappy, will release later (sorry)

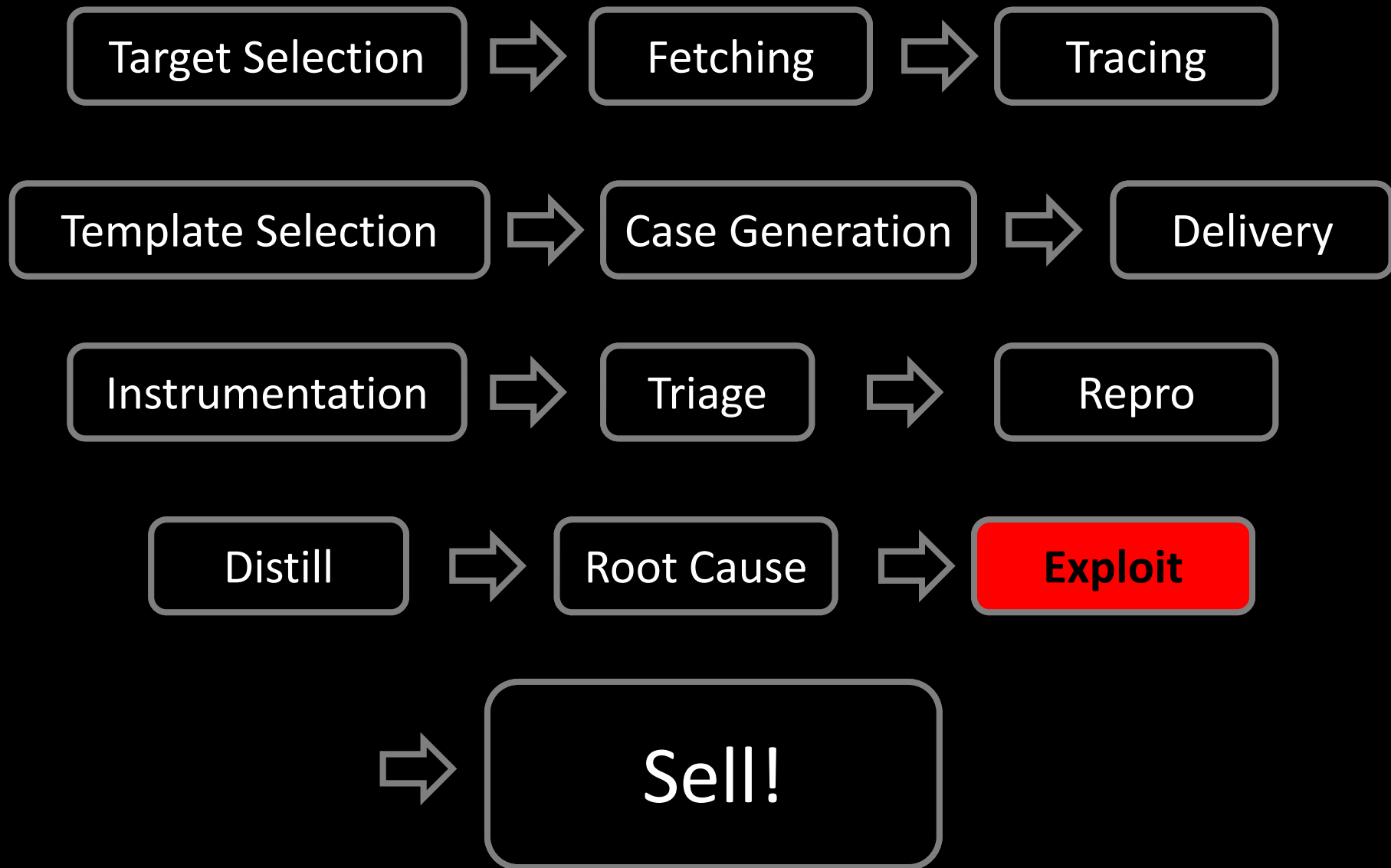


TraceDiff (+ demo)

- Presented very alpha code at BH USA 2010
- Trace orig and crashfile
- Compress as heirarchical grammars
- diff the grammars
- Look for new 'edges'
- Will release when updated and shiny

LESSONS LEARNED THE HARD WAY

NEVER BE AFRAID TO BE LAME



Exploit

- Requires humans
- Teams are better than individuals
- Exploit ninjas are expensive
- You can't skip this step.

LESSONS LEARNED THE HARD WAY

BUGS BUY BURGERS, EXPLOITS BUY CARS

Target Selection



Fetching



Tracing

Template Selection



Case Generation



Delivery

Instrumentation



Triage



Repro

Distill



Root Cause



Exploit



Sell!

LESSONS LEARNED THE HARD WAY

WE'RE NOT WRITING THIS BIT DOWN

Questions? (\$5)

ben@coseinc.com (\$2)

grugq@coseinc.com (\$10)